CREDIT CARD DEFAULT

Detailed Project Report (DPR)

- Devarshi Choudhury

Objective:

The objective of the Credit Card Default Prediction System is to develop a solution that can assess the likelihood of a credit card user defaulting on their payments. By analyzing various demographic and payment data, the model aims to provide financial institutions with valuable insights to make informed decisions regarding credit risk assessment.

Benefits:

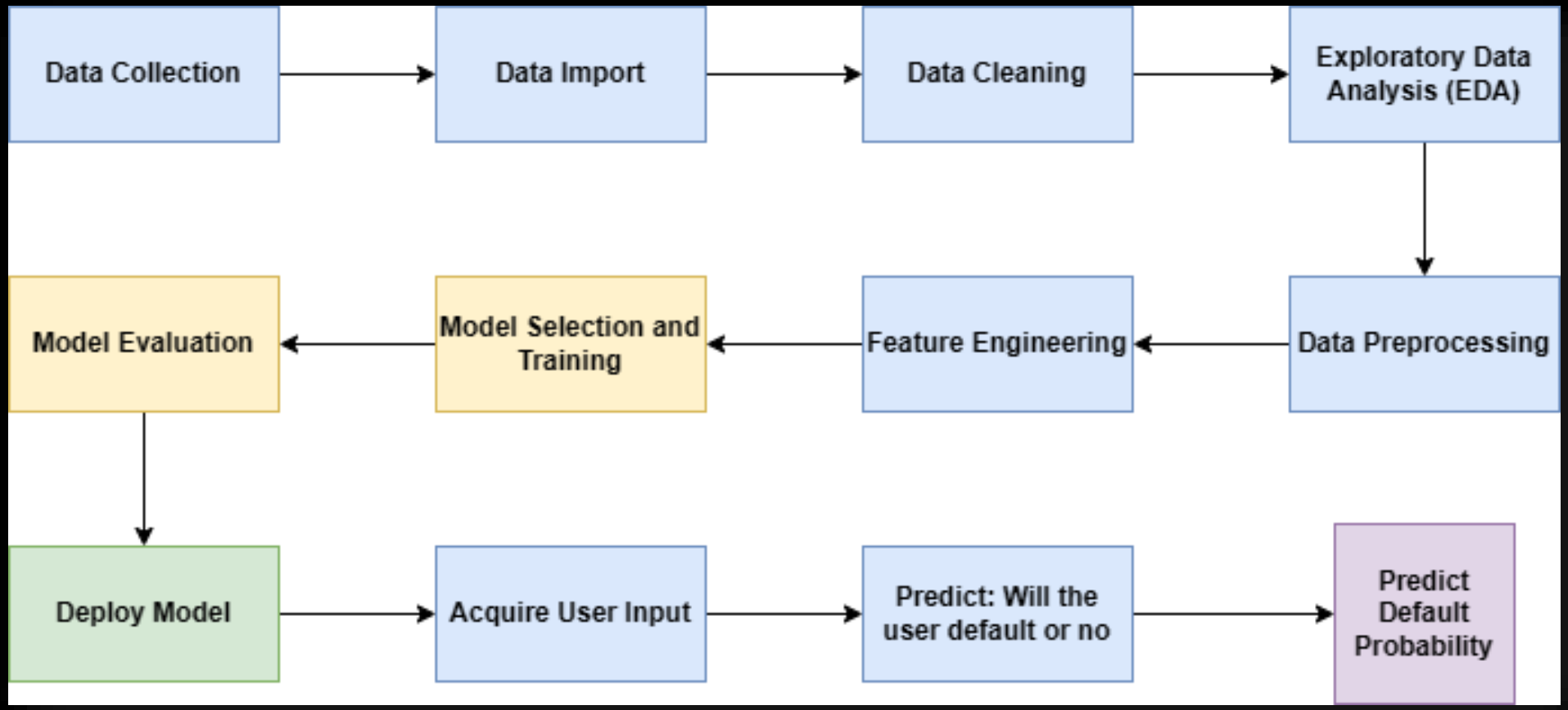
1. Risk Detection: The system helps in detecting potential credit defaults before they occur, allowing financial institutions to take proactive measures.

2. Improved Customer Insights: By analyzing demographic data such as gender, education level, marital status, and age, the system provides better insights into the customer base.

3. Efficient Resource Management: Financial institutions can allocate resources more effectively by focusing on customers with a higher probability of default.

4. Manual Intervention: If the system identifies a high probability of default, it alerts users, enabling manual inspection and further investigation into the customer's credit situation.

PROJECT ARCHITECTURE



DATASET INFORMATION

- This dataset, a part of the UCI Machine Learning Repository, has been obtained from Kaggle.
- Dataset Link: Default of Credit Card Clients Dataset (kaggle.com)

- **Exploratory Data Analysis (EDA):**

  1. Statistical Analysis:

  - Calculated basic statistics like mean, median, mode, standard deviation, etc., for numerical features such as credit limit, age, bill amounts, and payment amounts.
  - Evaluated the distribution of these numerical features to understand their central tendencies and variations.

  2. Visualization:

  - Created histograms to visualize the distribution of features like credit limit, age, bill amounts, and payment amounts.
  - Plotted bar charts to show the count of customers based on categorical features like gender, education, marital status, and repayment status.
  - Used box plots to identify outliers in numerical features.
  - Visualized the correlation matrix to understand the relationships between different features, especially the correlation with the target variable (default status).

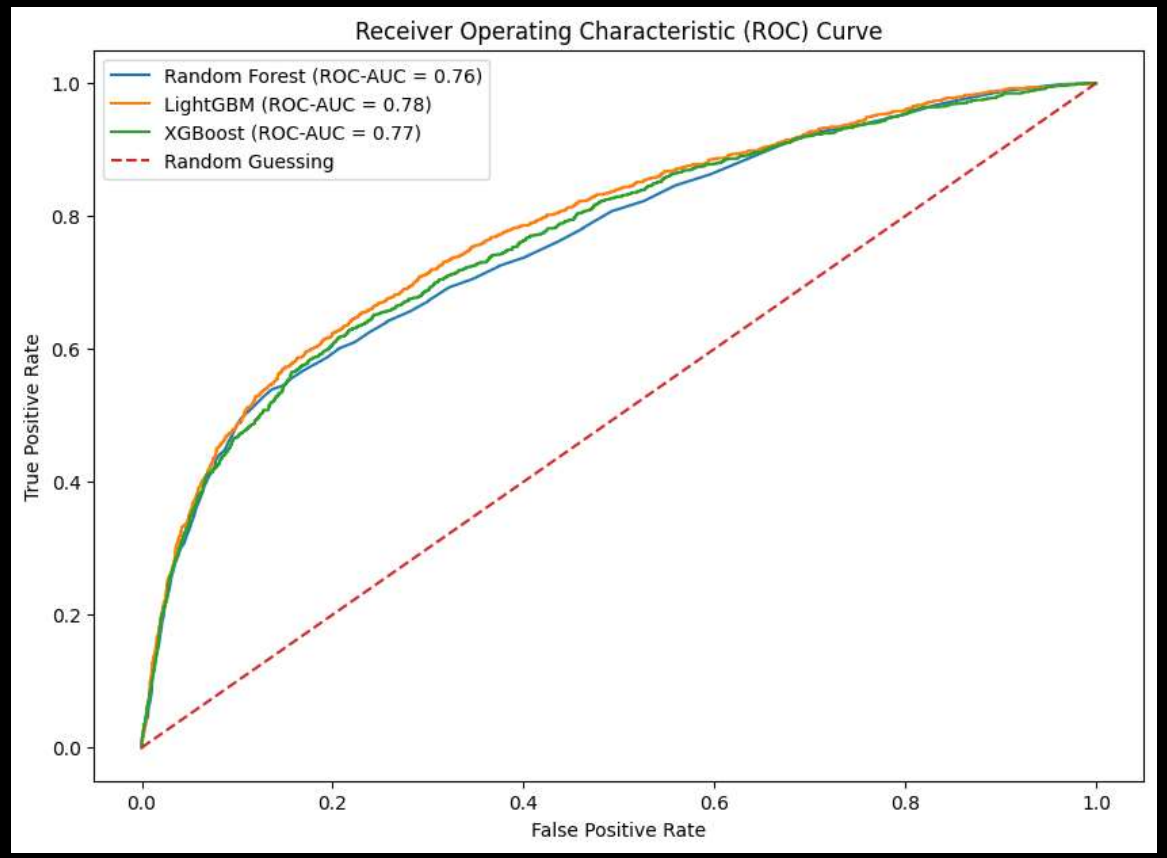
- **Data Cleaning and Preprocessing:**
  - Checked for missing and duplicate values in the dataset.
  - Checked and converted data types of features to the appropriate format for analysis.
  - Encoded categorical variables into numerical format using techniques like one-hot encoding or label encoding.
  - Checked for outliers using boxplot and Z-scores.

- **Feature Engineering and Selection:**
  - Created new features like 'Credit Utilization Ratio’ by combining bill amounts and credit limit.
  - Extracted relevant information from existing features to improve the predictive power of the model.
  - Derived pertinent insights from the engineered features to gain further understanding of the dataset.
  - Selected relevant features for model training using techniques like correlation analysis or feature importance from Trained models.
  - Eliminated less important or highly correlated features to reduce dimensionality and improve model efficiency.

- **Model Training and Evaluation:**
  - We selected multiple machine learning algorithms for training, including Random Forest, LightGBM, and XGBoost. Each of these algorithms was chosen for their strengths in handling classification tasks and their ability to handle the dataset's characteristics.
  - The dataset was split into training and testing sets using an 80-20 split. This ensured that the model was trained on a majority of the data while reserving a portion for evaluation.
  - Each selected algorithm was trained on the training data. This process involved feeding the algorithm with the input features (such as credit limit, gender, education, etc.) and the corresponding target variable (default or non-default).
  - After training, each model was evaluated using the testing set. Evaluation metrics such as accuracy, precision, recall, and F1-score were calculated to assess the model's performance in predicting credit card defaults.
  - To ensure the robustness of the models, k-fold cross-validation was performed. This technique splits the data into k subsets and trains the model k times, each time using a different subset for validation and the rest for training
  - Based on the evaluation results, the best-performing model was selected for deployment. credit card defaults.

BEST MODEL: LIGHTGBM (LIGHT GRADIENT BOOSTING MACHINE)

Achieved Results



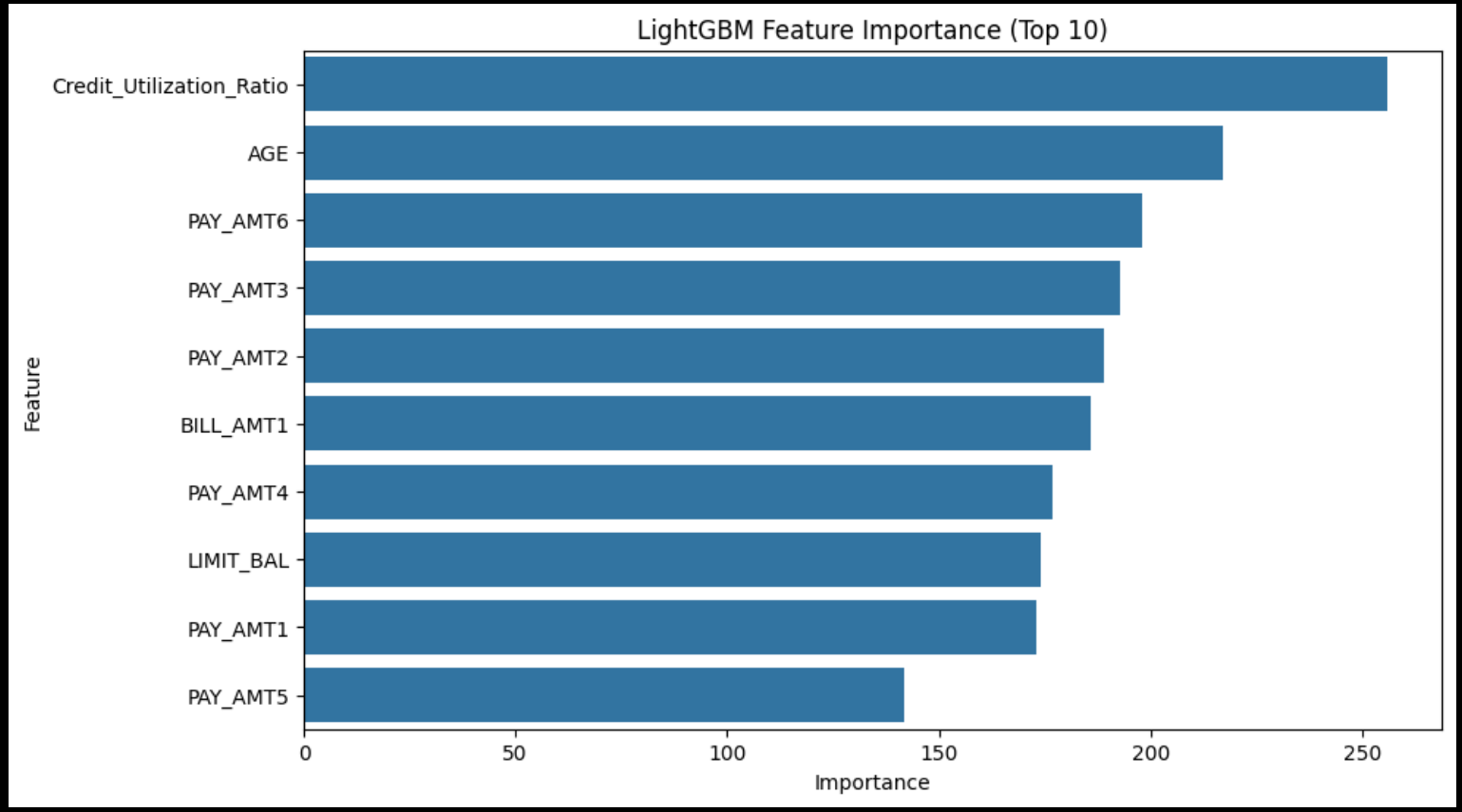

```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      4687
           1       0.66      0.36      0.46      1313

    accuracy                           0.82      6000
   macro avg       0.75      0.65      0.68      6000
weighted avg       0.80      0.82      0.80      6000

Accuracy: 0.8188333333333333
```

FEATURE IMPORTANCE OF LIGHTGBM



- **Webapp Development**
  - We designed the UI using Streamlit's layout features, creating sections for user input and displaying prediction results.
  - This involved using Streamlit's functions like st.sidebar for sidebar elements, st.subheader for section titles, st.number_input for numerical input fields, and st.selectbox for dropdowns.
  - We loaded a pre-trained LightGBM model using joblib’s load_model function, which loads the model from a .pkl file.
  - We defined a function predict_default to make predictions using the loaded model.
  - This function predicted whether a credit card user will default next month and provided the probability of default.
  - The result showed the predicted likelihood of default, credit utilization ratio, and the probability of default.
  - Additionally, we added an "**i**” Explanation button to provide a user-friendly explanation about the initial prediction and probability when the app first runs.
  - Finally, the app was deployed on Streamlit community Cloud.

DEPLOYMENT

- The frontend of the application was developed using Streamlit.
- The web application is deployed on streamlit community cloud.
- Committed the project to GitHub Repository
- Web application url: https://credit-card-default-prediction-lh653hvkh6akelqbfzzf5g.streamlit.app/

Amount of previous payments ($)

September PAY_AMT

0

June PAY_AMT

0

August PAY_AMT

0

May PAY_AMT

0

July PAY_AMT

0

April PAY_AMT

0

Prediction Result

Low likelihood of default.

Credit Utilization Ratio: 0.00%

Probability of Default: 11.30%

i

FREQUENTLY ASKED QUESTIONS (FAQS)

Q1) What's the source of data?

- The data is sourced from Kaggle.

Q 2) What was the type of data?

- The data was the combination of numerical and Categorical values.

Q 3) What's the complete flow you followed in this Project?

- Please refer slide number 3 to understand the project flow.

Q4) What techniques were you using for data pre-processing?

- Removing unwanted attributes
- Visualizing relation of independent variables with each other and output variables
- Creating new features such as 'Credit Utilization Ratio', 'Age Groups', and 'Payment Status Trends' were engineered from the existing attributes to check the impact on model performance.
- Feature Selection techniques like Variance Inflation Factor (VIF) analysis to detect and potentially remove highly correlated features.
- Cleaning data and checking for null values if present.
- Handled categorical variables by encoding them into numeric values.

Q 5) How Prediction was done?

- In our project, prediction was done using a trained machine learning model. The user inputs demographic and payment data, which is then preprocessed to prepare it for the model. The preprocessed data is fed into the trained Random Forest Classifier model. The model then predicts whether the user is likely to default on their credit card payment next month. The prediction result, along with the probability of default, is displayed to the user.